# Exercises: Functions, Triggers and Transactions

This document defines the **exercise assignments** for the ["Databases Basics - MSSQL" course @ Software University.](https://softuni.bg/trainings/1436/databases-basics-mssql-september-2016)

# Part I – Queries for SoftUni Database

## Employees with Salary Above 35000

Create storedprocedure **usp\_GetEmployeesSalaryAbove35000** that returns **all employees’ first and last names** for whose **salary is above 35000**. Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

|  |  |
| --- | --- |
| **First Name** | **Last Name** |
| Roberto | Tamburello |
| David | Bradley |
| Terri | Duffy |
| … | … |

## Employees with Salary Above Number

Create stored procedure **usp\_GetEmployeesSalaryAboveNumber** that **accept a number** (of type **MONEY**) as parameter and return **all employees’ first and last names** whose salary is **above or equal** to the given number. Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

Supplied number for that example is 48100.

|  |  |
| --- | --- |
| **First Name** | **Last Name** |
| Terri | Duffy |
| Jean | Trenary |
| Ken | Sanchez |
| … | … |

## Town Names Starting With

Write a stored procedure **usp\_GetTownsStartingWith** that **accept string as parameter** and returns **all town names starting with that string.** Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

Here is the list of all towns **starting with “b”.**

|  |
| --- |
| **Town** |
| Bellevue |
| Bothell |
| Bordeaux |
| Berlin |

## Employees from Town

Write a stored procedure **usp\_GetEmployeesFromTown** that accepts **town name** as parameter and return the **employees’ first and last name that live in the given town.** Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

Here it is a list of employees **living in Sofia.**

|  |  |
| --- | --- |
| **First Name** | **Last Name** |
| Svetlin | Nakov |
| Martin | Kulov |
| George | Denchev |

## Salary Level Function

Write a function **ufn\_GetSalaryLevel(@salary MONEY)** that receives **salary of an employee** and returns the **level of the salary**.

* If salary is **< 30000** return **“Low”**
* If salary is **between 30000 and 50000 (inclusive)** return **“Average”**
* If salary is **> 50000** return **“High”**

Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

|  |  |
| --- | --- |
| **Salary** | **Salary Level** |
| 13500.00 | Low |
| 43300.00 | Average |
| 125500.00 | High |

## Employees by Salary Level

Write a stored procedure **usp\_EmployeesBySalaryLevel** that receive as **parameter** **level of salary** (low, average or high) and print the **names of all employees** that have given level of salary. You can use the function - “**dbo.ufn\_GetSalaryLevel(@Salary)**”, which was part of the previous task, inside your “**CREATE PROCEDURE …**” query.

### Example

Here is the list of all employees with high salary.

|  |  |
| --- | --- |
| **First Name** | **Last Name** |
| Terri | Duffy |
| Jean | Trenary |
| Ken | Sanchez |
| … | … |

## Define Function

Define a function **ufn\_IsWordComprised(@setOfLetters, @word)** that returns **true** or **false** depending on that if the word is a comprised of the given set of letters. Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

|  |  |  |
| --- | --- | --- |
| **SetOfLetters** | **Word** | **Result** |
| oistmiahf | Sofia | 1 |
| oistmiahf | halves | 0 |
| bobr | Rob | 1 |
| pppp | Guy | 0 |

## \* Delete Employees and Departments

Write a SQL query to delete all Employees from the **Production** and **Production Control** departments. **Delete these departments from the Departments table** too. Submit your query as Run skeleton, run queries and check DB.After that exercise restore your database to revert those changes.

### Hint:

You may set **ManagerID** column in Departments table to **nullable** (using query "ALTER TABLE …").

## Employees with Three Projects

Create a procedure **usp\_AssignProject(@emloyeeId, @projectID)** that **assigns** **projects** to employee. If the employee has more than **3** project throw **exception** and **rollback** the changes. The exception message must be: "**The employee has too many projects!**" with Severity = 16, State = 1.

# PART II – Queries for Bank Database

## Find Full Name

You are given a database schema with tables **AccountHolders(Id (PK), FirstName, LastName, SSN)** and **Accounts(Id (PK), AccountHolderId (FK), Balance)**. Write a stored procedure **usp\_GetHoldersFullName** that selects the full names of all people. Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

|  |
| --- |
| **Full Name** |
| Susan Cane |
| Kim Novac |
| Jimmy Henderson |
| … |

## People with Balance Higher Than

Your task is to create a stored procedure **usp\_GetHoldersWithBalanceHigherThan** that accepts a **number as a parameter** and returns all **people who have more money in total of all their accounts than the supplied number**. Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

|  |  |
| --- | --- |
| **First Name** | **Last Name** |
| Susan | Cane |
| Petar | Kirilov |
| … | … |

## Future Value Function

Your task is to create a function **ufn\_CalculateFutureValue** that accepts as parameters – **sum (money)**, **yearly interest rate (float)** and **number of years(int)**. It should calculate and return the future value of the initial sum. Using the following formula:

* **I** – Initial sum
* **R** – Yearly interest rate
* **T** – Number of years

Submit your query statement as Run skeleton, run queries & check DB in Judge.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Initial sum:** 1000  **Yearly Interest rate:** 10%  **years:** 5  ufn\_CalculateFutureValue(1000, 0.1, 5) | 1610.51 |

## Calculating Interest

Your task is to create a stored procedure **usp\_CalculateFutureValueForAccount** that uses the function from the previous problem to give an interest to a person's account **for 5 years**, along with information about his/her **account id, first name, last name and current balance** as it is shown in the example below. It should take the **AccountId** and the **interest rate** as parameters. Again you are provided with “**dbo.ufn\_CalculateFutureValue**” function which was part of the previous task.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Account Id** | **First Name** | **Last Name** | **Current Balance** | **Balance in 5 years** |
| 1 | Susan | Cane | 123.12 | 198.286 |

\*Note: for the example above interest rate is 0.1

## Deposit Money

Add stored procedure **usp\_DepositMoney** (AccountId, moneyAmount) that operate in transactions. Submit your query statement as Run skeleton, run queries & check DB in Judge.

## Withdraw Money

Add stored procedures **usp\_WithdrawMoney** (AccountId, moneyAmount) that operate in transactions. Submit your query statement as Run skeleton, run queries & check DB in Judge.

## Money Transfer

Write stored procedure **usp\_TransferMoney**(senderId, receiverId, amount) that **transfers money from one account to another**. Consider cases when the amount of **money is negative number**. Make sure that the whole procedure **passes without errors** and **if error occurs make** **no change in the database.** You can use both: “**usp\_DepositMoney**”, “**usp\_WithdrawMoney**” (look at previous two problems about those procedures).

## Create Table Logs

Create another table – **Logs** (LogId, AccountId, OldSum, NewSum). Add a **trigger** to the Accounts table that **enters** a new entry into the **Logs** table every time the sum **on** an **account** **changes**.

**Submit** your query **only** for the **trigger** action as Run skeleton, run queries and check DB.

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **LogId** | **AccountId** | **OldSum** | **NewSum** |
| 1 | 1 | 123.12 | 113.12 |
| … | … | … | … |

## Create Table Emails

Create another table – **NotificationEmails**(Id, Recipient, Subject, Body). Add a **trigger** to logs table and **create new email whenever new record is inserted in logs table.** The following data is required to be filled for each email:

* **Recipient** – AccountId
* **Subject** – “Balance change for account: **{AccountId}**”
* **Body** - “On **{date}** your balance was changed from **{old}** to **{new}.**”

**Submit** your query **only** for the **trigger** action as Run skeleton, run queries and check DB.

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Recipient** | **Subject** | **Body** |
| 1 | 1 | Balance change for account: 1 | On Sep 12 2016 2:09PM your balance was changed from 113.12 to 103.12. |
| … | … | … | … |

# PART III – Queries for Diablo Database

You are given a **database "Diablo"** holding users, games, items, characters and statistics available as SQL script. Your task is to write some stored procedures, views and other server-side database objects and write some SQL queries for displaying data from the database.

**Important:** start with a **clean copy of the "Diablo" database** **on each problem**. Just execute the SQL script again.

## \*Scalar Function: Cash in User Games Odd Rows

Create a **function** **ufn\_CashInUsersGames** that **sums the cash of odd rows**.Rows must be ordered by cash in descending order. The function should take a game name as a parameter and **return the result as table**. Submit **only your function** **in judge** as Run skeleton, run queries & check DB.

Execute the function over the following game names, ordered exactly like: “**Bali**”, “**Lily Stargazer**”, “**Love in a mist**”, “**Mimosa**”, “**Ming fern**”.

### Output

|  |
| --- |
| **SumCash** |
| 55\*\*.\*\* |
| 72\*\*.\*\* |
| 85\*\*.\*\* |
| … |

### Hint

Use **ROW\_NUMBER** to get the rankings of all rows based on order criteria.

## Trigger

Users should not be allowed to buy items with higher level than their level. Create a trigger that restricts that. Trigger should prevent inserting items that are above specified level while allowing all others to be inserted.

Add bonus cash of 50000 to users: **baleremuda, loosenoise, inguinalself, buildingdeltoid, monoxidecos** in the game **“Bali”.**

There are two groups of **items** that you should buy for the above users in the game. First group is with **id between 251 and 299 including**. Second group is with **id between 501 and 539 including.**

**Take** off **cash** from each user **for** the bought **items**.

Select all users in the current game (“Bali”) with their items. Display **username**, **game name**, **cash** and **item name**. Sort the result by username alphabetically, then by item name alphabetically.

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Username** | **Name** | **Cash** | **Item Name** |
| baleremuda | Bali | 4\*\*\*\*.\*\* | Iron Wolves Doctrine |
| baleremuda | Bali | 4\*\*\*\*.\*\* | Irontoe Mudsputters |
| … | … | … | … |
| buildingdeltoid | Bali | 3\*\*\*\*.\*\* | Alabaster Gloves |
| … | … | … | … |

## \*Massive Shopping

1. User **Stamat** in **Safflower** gamewants to buy some items. He likes all items **from Level 11 to 12** as well as all items **from Level 19 to 21.** As it is a bulk operation you have to **use transactions.**
2. A transaction is the operation of taking out the cash from the user in the current game as well as adding up the items.
3. Write transactions for each level range. If anything goes wrong turn back the changes inside of the transaction.
4. Extract all item names in the given game sorted by name alphabetically

Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |
| --- |
| **Item Name** |
| Akarats Awakening |
| Amulets |
| Angelic Shard |
| … |

## Number of Users for Email Provider

Find number of users for email provider from the largest to smallest, then by Email Provider in ascending order. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |
| --- | --- |
| **Email Provider** | **Number Of Users** |
| yahoo.com | 14 |
| dps.centrin.net.id | 5 |
| softuni.bg | 5 |
| indosat.net.id | 4 |
| … | … |

## All User in Games

Find all **user in games** with information about them. Display the game name, game type, username, level, cash and character name. Sort the result by level in descending order, then by username and game in alphabetical order. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Game** | **Game Type** | **Username** | **Level** | **Cash** | **Character** |
| Calla lily white | Kinky | obliquepoof | 99 | 7527.00 | Monk |
| Dubai | Funny | rateweed | 99 | 7499.00 | Barbarian |
| Stonehenge | Kinky | terrifymarzipan | 99 | 4825.00 | Witch Doctor |
| … |  | … | … | … | … |

## Users in Games with Their Items

Find all users in games with their items count and items price. Display the username, game name, items count and items price. Display only user in games with items count more or equal to 10. Sort the results by items count in descending order then by price in descending order and by username in ascending order. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Username** | **Game** | **Items Count** | **Items Price** |
| skippingside | Rose Fire & Ice | 23 | 11065.00 |
| countrydecay | Star of Bethlehem | 18 | 8039.00 |
| obliquepoof | Washington D.C. | 17 | 5186.00 |

## \* User in Games with Their Statistics

Find information about **every game** a user has played with their **statistics**. Each user may have participated in several games. Display the username, game name, character name, strength, defence, speed, mind and luck.  
Every statistic (strength, defence, speed, mind and luck) should be a sum of the **character statistic, game type statistic** and **items** for user **in game** statistic. One user may have multiple characters in a single game. What you should do in order to calculate the statistic properly is to sum the following:

1. Get the sum of all items - of all characters that the user may have(SUM).
2. For all of his characters get the character stats which are the biggest (MAX).
3. For all of his game types stats select only these which are again the biggest (MAX).

Order the results by **Strength**, **Defence**, **Speed**, **Mind**, **Luck** – all in descending order. Submit your query statement as Prepare DB & run queries in Judge.

### Example

Let’s say that we have user “**Ana**” and she is in the game “**Star of Bethlehem**” having two characters: Sorceress and Paladin. In tables below will be given their statistics:

**Paladin**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type of Stats\Statistics** | **Strength** | **Defence** | **Speed** | **Mind** | **Luck** |
| Item A Stats | 15 | 10 | 3 | 14 | 20 |
| Game Type Stats | 5 | 5 | 7 | 4 | 5 |
| Character Stats | 20 | 17 | 10 | 8 | 6 |

**Sorceress**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type of Stats\Statistics** | **Strength** | **Defence** | **Speed** | **Mind** | **Luck** |
| Item B Stats | 8 | 4 | 10 | 22 | 12 |
| Game Type Stats | 6 | 6 | 6 | 4 | 6 |
| Character Stats | 8 | 6 | 13 | 23 | 10 |

What we should get as a result is:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Username** | **Game** | **Character** | **Strength** | **Defence** | **Speed** | **Mind** | **Luck** |
| Ana | Star of Bethlehem | Sorceress | 49 | 37 | 33 | 63 | 48 |

Now let’s see how the Strength is calculated:

Strength – (Item A’s Strength + Item B’s Strength) + MAX (Paladin Game Type’s Strength, Sorceress GameType's Strength) + MAX (Paladin Character’s Strength, Sorceress Character’s Strength) = 15 + 8 + 6 + 20 = 49.

Here we sum up first the **items** **stats** (**15 + 8**) then we add the biggest one between the game type stats (6 > 5 => **6**) then we add the biggest one between the character stats (20 > 8 => **20**). So (15 + 8) + 6 + 20 = 49.

Same goes for the Luck:

Luck = (Item A’s Luck + Item B’s Luck) + MAX (Paladin Game Type’s Luck, Sorceress GameType's Luck) + MAX (Paladin Character’s Luck, Sorceress Character’s Luck) = 20 + 12 + 6 + 10 = 49.

Here we sum up first the **items** **stats** (**20 + 12**) then we add the biggest one between the game type stats (6 > 5 => **6**) then we add the biggest one between the character stats (10 > 6 => **10**). So (20 + 12) + 6 + 10 = 48.

### Output

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Username** | **Game** | **Character** | **Strength** | **Defence** | **Speed** | **Mind** | **Luck** |
| skippingside | Rose Fire & Ice | Sorceress | 258 | 215 | 246 | 240 | 263 |
| countrydecay | Star of Bethlehem | Sorceress | 221 | 163 | 216 | 153 | 196 |
| obliquepoof | Washington D.C. | Paladin | 204 | 200 | 183 | 185 | 185 |

## All Items with Greater than Average Statistics

Find all items with statistics larger than average. Display only items that have **Mind, Luck** and **Speed** greater than average **Items** mind, luck and speed. Sort the results by item names in alphabetical order. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Price** | **MinLevel** | **Strength** | **Defence** | **Speed** | **Luck** | **Mind** |
| Aether Walker | 473.00 | 46 | 2 | 10 | 15 | 11 | 13 |
| Ancient Parthan Defenders | 566.00 | 38 | 5 | 7 | 10 | 19 | 18 |
| Aquila Cuirass | 405.00 | 76 | 5 | 7 | 10 | 19 | 18 |
| Arcstone | 613.00 | 50 | 2 | 10 | 15 | 11 | 13 |

## Display All Items with Information about Forbidden Game Type

Find **all** **items** and information whether and what forbidden game types they have. Display item name, price, min level and forbidden game type. Display all items. Sort the results by game type in descending order, then by item name in ascending order. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Price** | **MinLevel** | **Forbidden Game Type** |
| Archfiend Arrows | 531.00 | 8 | Kinky |
| Behistun Rune | 611.00 | 67 | Kinky |
| Boots | 782.00 | 44 | Kinky |
| … | … | … | … |

## Buy Items for User in Game

1. User **Alex** isin theshop in the game “**Edinburgh**” and she wants to buy some items. She likes **Blackguard**, **Bottomless Potion of Amplification**, **Eye of Etlich (Diablo III)**, **Gem of Efficacious Toxin**, **Golden Gorget of Leoric** and **Hellfire Amulet**. Buy the items. You should add the data in the right tables. Get the money for the items from user in game **Cash**.
2. Select all users in the current game with their items. Display username, game name, cash and item name. Sort the result by item name.

Submit your query statements as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Username** | **Name** | **Cash** | **Item Name** |
| Alex | Edinburgh | \*\*\*\*.\*\* | Akanesh, the Herald of Righteousness |
| … | … | … | … |
| corruptpizz | Edinburgh | \*\*\*\*.\*\* | Broken Crown |
| … | … | … | … |
| printerstencils | Edinburgh | \*\*\*\*.\*\* | Envious Blade |

# PART IV – Queries for Geography Database

## Peaks and Mountains

Find all **peaks along with their mountain** sorted by elevation (from the highest to the lowest), then by peak name alphabetically. Display the peak name, mountain range name and elevation. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |
| --- | --- | --- |
| **PeakName** | **Mountain** | **Elevation** |
| Everest | Himalayas | 8848 |
| K2 | Karakoram | 8611 |
| Kangchenjunga | Himalayas | 8586 |
| … |  | … |

## Peaks with Their Mountain, Country and Continent

Find all peaks along with their mountain, country and continent. When a mountain belongs to multiple countries, display them all. Sort the results by peak name alphabetically, then by country name alphabetically. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **PeakName** | **Mountain** | **CountryName** | **ContinentName** |
| Aconcagua | Andes | Argentina | South America |
| Aconcagua | Andes | Chile | South America |
| Banski Suhodol | Pirin | Bulgaria | Europe |
| … | … | … | … |

## Rivers by Country

For each country in the database, display the number of rivers passing through that country and the total length of these rivers. When a country does not have any river, display **0** as rivers count and as total length. Sort the results by rivers count (from largest to smallest), then by total length (from largest to smallest), then by country alphabetically. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **CountryName** | **ContinentName** | **RiversCount** | **TotalLength** |
| China | Asia | 8 | 35156 |
| Russia | Europe | 6 | 26427 |
| … |  | … | … |

## Count of Countries by Currency

Find the **number of countries for each currency**. Display three columns: currency code, currency description and number of countries. Sort the results by number of countries (from highest to lowest), then by currency description alphabetically. Name the columns exactly like in the table below. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |
| --- | --- | --- |
| **CurrencyCode** | **Currency** | **NumberOfCountries** |
| EUR | Euro Member Countries | 35 |
| USD | United States Dollar | 17 |
| AUD | Australia Dollar | 8 |
| XOF | Communauté Financière Africaine (BCEAO) Franc | 8 |
| … | … | … |

## Population and Area by Continent

For each continent, display the total area and total population of all its countries. Sort the results by population from highest to lowest. Submit your query statement as Prepare DB & run queries in Judge.

### Output

|  |  |  |
| --- | --- | --- |
| **ContinentName** | **CountriesArea** | **CountriesPopulation** |
| Asia | 31603228 | 4130318467 |
| Africa | 30360296 | 1015470588 |
| … | … | … |

## Monasteries by Country

1. Create a **table Monasteries(Id, Name, CountryCode)**. Use auto-increment for the primary key. Create a **foreign key** between the tables Monasteries and Countries.
2. Execute the following SQL script (it should pass without any errors):

|  |
| --- |
| INSERT INTO Monasteries(Name, CountryCode) VALUES  ('Rila Monastery “St. Ivan of Rila”', 'BG'),  ('Bachkovo Monastery “Virgin Mary”', 'BG'),  ('Troyan Monastery “Holy Mother''s Assumption”', 'BG'),  ('Kopan Monastery', 'NP'),  ('Thrangu Tashi Yangtse Monastery', 'NP'),  ('Shechen Tennyi Dargyeling Monastery', 'NP'),  ('Benchen Monastery', 'NP'),  ('Southern Shaolin Monastery', 'CN'),  ('Dabei Monastery', 'CN'),  ('Wa Sau Toi', 'CN'),  ('Lhunshigyia Monastery', 'CN'),  ('Rakya Monastery', 'CN'),  ('Monasteries of Meteora', 'GR'),  ('The Holy Monastery of Stavronikita', 'GR'),  ('Taung Kalat Monastery', 'MM'),  ('Pa-Auk Forest Monastery', 'MM'),  ('Taktsang Palphug Monastery', 'BT'),  ('Sümela Monastery', 'TR') |

1. Write a SQL command to add a new Boolean column IsDeleted in the Countries table (defaults to false). Note that there is no "Boolean" type in SQL server, so you should use an alternative and make sure you set the **default** value properly.
2. Write and execute a SQL command to **mark as deleted all countries that have more than 3 rivers**.
3. Write a query to display all **monasteries** along with their **countries** sorted by monastery name. Skip all deleted countries and their monasteries.

Submit your query statements **only for subtasks 1, 2, 4 and 5** **at once** as Prepare DB & run queries in Judge.

### Output

|  |  |
| --- | --- |
| **Monastery** | **Country** |
| Bachkovo Monastery “Virgin Mary” | Bulgaria |
| Benchen Monastery | Nepal |
| Kopan Monastery | Nepal |
| … | … |

## Monasteries by Continents and Countries

This problem assumes that **the previous problem is completed successfully without errors**.

1. Write and execute a SQL command that **changes the country named "Myanmar" to its other name "Burma"**.
2. Add a **new monastery** holding the following information: Name="Hanga Abbey", Country="Tanzania".
3. Add a **new monastery** holding the following information: Name="Myin-Tin-Daik", Country="Myanmar".
4. Find the **count of monasteries for each continent and not deleted country**. Display the **continent name**, the **country name** and the **count of monasteries**. Include also the countries with 0 monasteries. Sort the results by monasteries count (from largest to lowest), then by country name alphabetically. Name the columns exactly like in the table below.

Submit all your query statements at once as Prepare DB & run queries in Judge.

\* Note when you **insert** the **monasteries** make sure to specify the country code by the country name (aka use   
 **subquery**).

### Output

|  |  |  |
| --- | --- | --- |
| **ContinentName** | **CountryName** | **MonasteriesCount** |
| Asia | Nepal | 4 |
| Europe | Bulgaria | 3 |
| Asia | Burma | 2 |
| Europe | Greece | 2 |
| … | … | … |